ABC INFORMATIQUE

N° 96

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

L'intelligence des robots

Le micro qui lit

Un docteur pour le PC

Drames en série

EDITIONS ATLAS

M 6062-96-12,50 F 90 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

Lueur d'intelligence

La mise au point effective de robots intelligents se heurte encore à de très nombreux problèmes. Voyons d'un peu plus près le fossé entre la théorie et la pratique.

Robyn Beeche

Starlette cybernétique

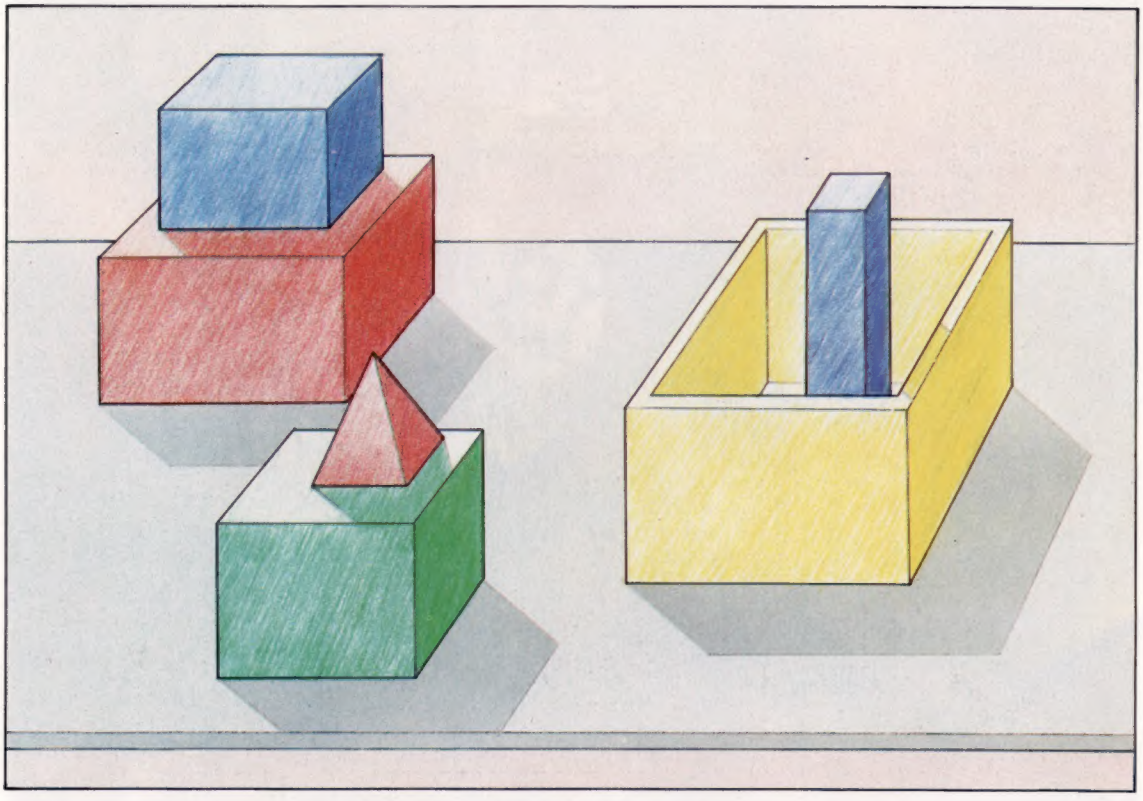
Dès 1966, Bruce Lacey (qu'on voit ici à côté de sa création) mit au point Rosa Bosom afin de lui faire jouer le rôle de la reine de France dans une adaptation théâtrale des *Trois Mousquetaires* représentée au Royal Court Theatre de Londres. Construite à l'aide de relais et de moteurs achetés dans des surplus militaires, Rosa obéit à des commandes qui prennent la forme de signaux sonores, qu'elle convertit en mouvements particuliers. Elle est aussi dotée de capteurs à ultrasons qui lui permettent de détecter et d'éviter les obstacles. Sa « voix » était en fait enregistrée sur une bande magnétique, tandis que ses lèvres remuaient par télécommande. Encore plus intéressant : elle pouvait évoluer de concert avec un second robot baptisé Mate.
Rosa a été présentée dans de nombreuses expositions, et a même gagné un concours de beauté !

De tous les domaines de l'informatique, la robotique est sans doute celui où l'écart entre l'imagination et la réalité est le plus grand. Hollywood n'est pas avare de robots de science-fiction capables de marcher, de parler, et même de comploter la destruction de l'humanité ! Mais, non loin de là, à l'université de Stanford ou à Toulouse, en Europe, les modèles les plus sophistiqués peuvent à peine traverser une pièce un tant soit peu encombrée sans se cogner aux meubles.

Bien entendu, il existe déjà des robots incapables de reconnaître des formes, installés le long des chaînes de montage, dans l'industrie automobile notamment, qui accomplissent des tâches de soudure ou de peinture. Mais ils se bornent à répéter des séquences d'opérations préprogrammées, sans jamais varier d'un pouce. Ils sont si dépourvus d'intelligence que toute interruption imprévue les voit s'agiter dans le vide, sans se rendre compte de rien...

Cet abîme entre la théorie et la pratique a des causes très complexes que l'on peut résumer en deux mots : « perception » et « planification ». Les robots actuels sont incapables d'analyser les données transmises par leurs capteurs. S'ils le pouvaient, ils ne sauraient d'ailleurs pas quoi faire des résultats. C'est bien pourquoi la robotique est un terrain d'expérimentation privilégié pour les théories relatives à l'intelligence artificielle. Le roboticien s'efforce en effet de mettre au point un appareil capable d'affronter le monde réel, et pour cela il faut impérativement que le dispositif en question soit au moins en mesure d'analyser son environnement.

Dans tous les autres domaines, ou presque, de l'intelligence artificielle, le programmeur peut très bien se limiter à un « micro-monde » qu'il a créé lui-même. Un logiciel de jeu d'échecs opère au sein d'un environnement parfaitement abstrait. Les systèmes experts s'intéressent aux faits et non aux choses. Aussi les chercheurs sont-ils toujours tentés de se limiter à un « monde de blocs ».



Un monde artificiel
Le « monde des blocs » est en fait un modèle très abstrait du monde réel. Les chercheurs en intelligence artificielle s'en servent pour étudier l'analyse de scènes, la compréhension des messages et la résolution des problèmes. Mais les réussites, bien réelles, obtenues par ce moyen se révèlent très difficiles à adapter dans la réalité.
(Cl. Kevin Jones.)

Il n'est pas très difficile de connecter un robot à toute une batterie d'appareils — caméras de télévision, détecteurs de chaleur, palpeurs, dispositifs à ultrasons, etc. — qui pourront lui fournir des données inaccessibles aux humains : la lumière infrarouge ou ultraviolette par exemple. Mais, à moins de se déplacer dans un univers très étroitement défini, l'engin ne comprendra pas le sens des informations qu'il reçoit.

Mettre au point un algorithme efficace qui permettra de trouver un chemin à travers un labyrinthe est une chose. S'en servir pour guider un robot dans un environnement urbain en est une autre : il faut tenir compte de tous les obstacles imaginables qu'il peut rencontrer.

Rappelons les raisons du grand succès des compétitions comme « Micromouse ». Une « souris » contrôlée par ordinateur doit s'orienter, le plus rapidement possible, au sein d'un labyrinthe, de façon à parvenir au centre en un temps très court. La théorie donne les principaux algorithmes pour ce faire. Mais les murs ne sont jamais parfaitement parallèles; il y a, de-ci de-là, des flaques d'huile dues aux concurrents précédents, et ainsi de suite. Dans ces conditions, la robustesse de l'engin et ses facultés d'adaptation auront bien plus d'importance que l'algorithme qui le fait se mouvoir.

Il s'agit en fait d'un environnement très simplifié, réduit à quelques cubes semblables à ceux des enfants. Bien entendu, il ne s'agit pas de cubes « réels », mais de leur représentation hautement formalisée. Il est très agréable d'écrire des programmes qui peuvent résoudre des problèmes et manipuler des objets au sein d'un univers qui se réduit à l'unité centrale d'un ordinateur.

Un modèle cognitif

Le roboticien, malheureusement, ne travaille pas dans des conditions aussi faciles. Un appareil « intelligent » — c'est-à-dire capable de se déplacer de sa propre initiative — doit obligatoirement disposer d'un modèle cognitif de son environnement. Il doit également pouvoir réagir aux messages parlés. Comme il est impossible de les prévoir tous, il faudra donc qu'il puisse apprendre. Bref, les problèmes à résoudre sont énormes et loin d'être résolus. On peut les classer en quatre grandes catégories :

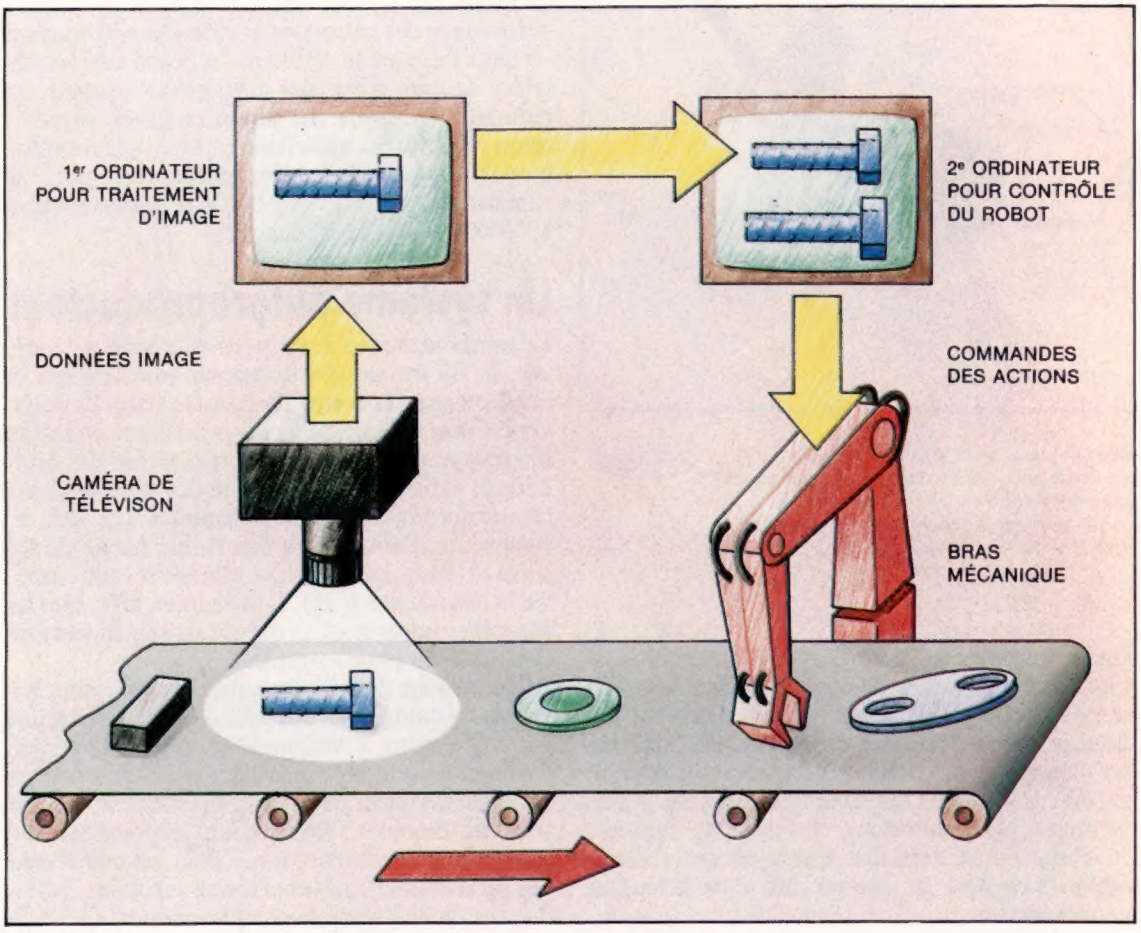
— vision informatisée;
— reconnaissance de la parole;
— résolution de problèmes dans un environnement dynamique;
— capacité d'apprentissage.

Encore n'est-ce qu'une liste réduite au minimum.

Les robots industriels modernes, par exemple, sont prévus pour évoluer au sein d'un environnement extrêmement structuré. Ce sont en fait de simples dispositifs mécaniques commandés par ordinateur et intégrés dans une chaîne de montage totalement automatisée. Leurs « mains » seront ainsi capables de visser des boulons, et un opérateur humain pourra leur « enseigner » des séquences de mouvements à répéter. Mais les robots ne peuvent que les mémoriser, sans jamais être en mesure de les modifier. Ce n'est d'ailleurs pas utile — ils travaillent dans un cadre si contraignant qu'ils n'ont aucun besoin de capacités de perception.

Pourtant certains robots sont d'ores et déjà capables de s'aventurer dans le monde extérieur. Ils sont un peu moins dépendants des humains, ou d'autres machines, et peuvent, de façon encore limitée, définir leurs propres tâches.

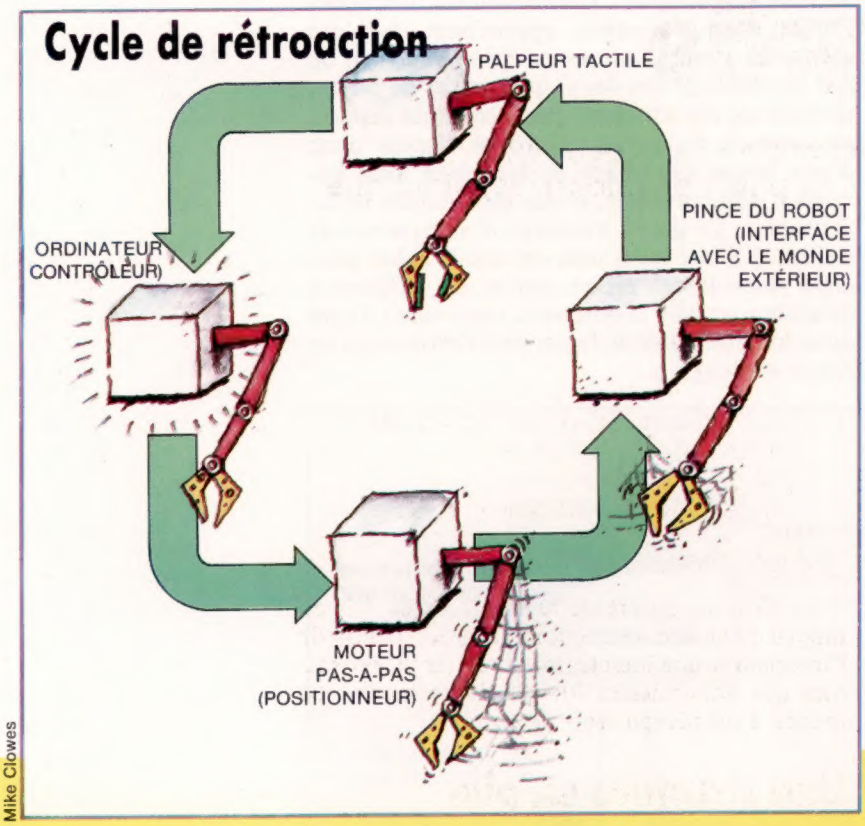
Un des moyens les plus simples pour doter le robot de perception est évidemment de le faire « voir ». Une caméra vidéo recueille des images (de l'objet qu'il doit prendre, par exemple) et les transmet à un ordinateur de contrôle (qui peut d'ailleurs être différent de celui qui commande le bras mécanique). L'information fait alors l'objet d'un traitement approfondi (par des méthodes « ascendantes », « descendantes », ou par un mélange des deux) qui permet de mettre en évidence des structures pertinentes par rapport au contexte de travail. Le robot dispose donc d'une image des objets et des outils avec lesquels il doit travailler, et des instructions indispensables. Le grand avantage d'un système de vision, si limité qu'il soit, est que le robot peut ainsi recevoir des pièces venues de différentes directions, et faire la différence entre elles : il peut donc les manipuler de façon plus correcte qu'un robot « aveugle ».



Des idées à la chaîne
Un robot capable de « voir » pourrait être mis en œuvre afin de reconnaître et de prendre des objets circulant sur une chaîne de montage. Il lui faudrait sans doute faire usage de deux ordinateurs : le premier serait chargé d'interpréter les données transmises par une caméra de télévision, et le second contrôlerait le bras mécanique lui-même et la saisie des objets. (Cl. Kevin Jones.)

Les concepteurs peuvent aussi doter le robot du sens du toucher, en plaçant des palpeurs au bout de ses « mains ». Lorsqu'ils entreront en contact avec un objet quelconque, l'information sera transmise à l'ordinateur de contrôle, qui se chargera de l'analyser. Si par exemple le robot vient à heurter un mur, non seulement la collision sera détectée, mais de surcroît l'erreur se verra étudiée de près, pour faire en sorte qu'elle ne se reproduise pas. Un test désormais classique consiste à faire percer un trou au robot : d'abord dans une pièce d'acier, puis dans un œuf... sans briser la coquille!

La robotique n'a encore mis au point que des systèmes de vision et de toucher assez rudimentaires. Elle se heurte en effet au problème de la rétroaction. Mais il est très difficile de donner à l'information venue du monde extérieur une



Mike Clowes

Quand nous prenons quelque chose, nous en estimons le poids afin de déterminer intuitivement quelle force nous allons devoir employer pour arriver à le soulever. Il s'agit essentiellement d'un cycle de rétroaction : des capteurs tactiles fournissent des données à l'ordinateur de contrôle, qui en retour augmente ou diminue la prise du bras sur l'objet en question.

forme acceptable (et surtout réellement utilisable) par l'ordinateur de contrôle. Une des grosses difficultés n'est autre que le traitement des données « en temps réel » : le robot devrait pouvoir réagir à un événement quelconque avant même qu'il ait pris fin. Quelques modèles y parviennent plus ou moins, à l'aide de capteurs et d'une petite capacité d'analyse de la rétroaction : ce sont les robots dits « de deuxième génération ».

Nous avons déjà vu que comprendre un message parlé continu était extrêmement difficile pour un système informatisé. Le principal obstacle en ce sens vient du fait que le robot n'a aucun contrôle sur ce qu'on peut lui dire — précisément parce qu'il doit pouvoir tout entendre ! Il est pourtant possible de lui donner un degré suffisant de reconnaissance de la parole (ce qu'on ne doit pas confondre avec la compréhension de celle-ci). En fait, cela se réduit à lui apprendre quelques mots ou quelques phrases, auxquels il devra réagir de façon appropriée, un peu comme un chien réagit à des ordres comme « Assis ! », « Couché ! » ou « Donne la patte ! ».

La recherche en robotique a beaucoup profité des programmes d'exploration spatiale. Les ingénieurs de la NASA ont par exemple mis au point un véhicule d'exploration baptisé Mars Rover, afin qu'il puisse évoluer à la surface de la planète Mars. Il sera équipé de détecteurs (caméras de télévision, en particulier) lui permettant de collecter des informations, et aura assez d'« intelligence » pour en faire réellement usage. Il serait en effet absolument inopérant de lui envoyer des signaux depuis la Terre à chaque fois qu'il doit entreprendre une nouvelle action : l'aller et retour radio prend environ une heure et demie... C'est un appareil qui ne sera mis en œuvre que dans quelques années, mais les Américains ont réussi dès 1976 à poser sur Mars deux sondes Viking. Elles se sont livrées à des analyses chimiques du sol et de l'atmosphère, sans être continuellement suivies par la station de contrôle de Houston.

Envoyer des robots — et non des astronautes — dans l'espace se révèle par ailleurs moins coûteux, ce qui n'est pas négligeable quand on connaît l'énormité des sommes mises en jeu : selon des sources américaines, lancer un homme dans l'espace, l'y faire demeurer, puis le ramener sain et sauf sur la Terre coûte environ 100 000 francs de l'heure...

Un système autoreproducteur

Le mathématicien John von Neumann a lancé, vers la fin des années quarante, une idée qui se révélera peut-être très fructueuse dans l'avenir. On lui doit en particulier les principaux concepts théoriques qui ont permis la réalisation des ordinateurs numériques, ainsi que des réflexions sur l'autoreproduction des automates (ce qui ne représente d'ailleurs qu'une faible partie de ses travaux, au premier rang desquels il faut compter la théorie des jeux). Il pensait en effet que les machines pourraient se reproduire en suivant un ensemble de règles simples.

Un système de ce genre aurait besoin, selon lui, de quatre composants de base. Tout d'abord une sorte d'« usine » automatisée qui pourrait rassembler des matières premières et les transformer en produits finis selon les instructions qui lui seraient données. Ensuite un duplicateur qui recopierait ces instructions, puis un contrôleur qui lui transmettrait les ordres à recopier. Enfin, il y aurait les instructions elles-mêmes, qui indiqueraient au système comment construire une nouvelle « usine » à partir des produits qu'il vient de fabriquer.

Ces considérations sont restées à l'état de spéculations théoriques pendant près de quarante ans, jusqu'à ce que les chercheurs de la NASA définissent un projet analogue en vue de la création d'un système autoreproducteur qui évoluerait sur la Lune. Cette usine spatiale comprendrait un « constructeur universel » qui se servirait des pièces créées par l'unité de production pour construire avec elles une nouvelle usine — qui serait également dotée d'un autre constructeur universel ! L'ensemble devrait recourir aux matières premières disponibles sur la Lune, et n'aurait pas besoin d'être suivi de la Terre. Il pourrait par ailleurs créer à son propre usage de nouveaux robots chargés d'effectuer des tâches particulières.

Début de partie

Définissons les modules qui nous permettront de jouer au go. Nous ferons usage d'une structure « en pile » des données, ainsi que du concept de récursivité.

Quand on doit rédiger un programme de ce type, il est nécessaire de recourir à plusieurs sous-programmes d'emploi général, dont des procédures de mise à jour du plateau de jeu, de vérification de la légalité de chaque coup, etc. C'est ainsi qu'un logiciel d'échecs aura forcément besoin d'un module qui verra si le roi est, ou non, soumis à une attaque.

Il est bien entendu tout à fait possible d'en rédiger un chargé de trouver les coups qui mettent le roi adverse en échec, et qui par ailleurs défend le nôtre de façon qu'il ne soit pas menacé. Il est cependant bien plus judicieux de mettre au point un sous-programme à valeur générale, qui répond par VRAI ou FAUX à la question : « Le roi de la couleur considérée est-il en échec? » Il suffira ensuite de lui donner à chaque fois une valeur différente pour la couleur, et de voir à quelles conclusions il aboutit.

Nous suivrons une démarche de ce type pour notre programme de go. Plutôt que d'additionner des modules consacrés chacun à un type de groupe particulier, nous nous servirons d'un sous-programme qui fait le décompte de toutes les pierres de chaque groupe. Nous pourrons ensuite, en appelant telle ou telle partie du programme, estimer l'importance réelle des résultats ainsi obtenus.

La version destinée au BBC Micro fait usage d'une procédure qui recevra le nom de PROC-recherche. Elle se voit transmettre deux paramètres : la position de chaque pierre du groupe considéré (P%), et la couleur de ce groupe (C%). Elle ajoute alors 1 à CSTN%, qui représente le nombre total de pierres du groupe, puis passe en revue les quatre pierres adjacentes. Pour chacune d'elles, la même procédure est répétée, à condition bien sûr qu'elle n'ait pas déjà été décomptée. Comme nous disposons déjà d'une procédure chargée d'effectuer cette recherche dans les quatre directions à partir d'une pierre, le plus simple est évidemment de faire en sorte qu'elle s'appelle elle-même! C'est un exemple de ce qu'on appelle la récurrence. Comme le concept peut paraître un peu compliqué à première vue, rendons les choses plus claires en décomposant la structure d'ensemble de la procédure de recherche :

```
RECHERCHE (position départ, couleur donnée)
 IF position de départ sur plateau
   AND position départ occupée par pierre de la couleur donnée
   AND position départ non encore décomptée
 THEN
   marquer position départ pour ne pas la décompter
   ultérieurement
   ajouter 1 à CSTN% (variable de comptage)
   RECHERCHE (au nord position départ, couleur donnée)
   RECHERCHE (à l'est, couleur donnée)
   RECHERCHE (au sud, couleur donnée)
   RECHERCHE (à l'ouest, couleur donnée)
 ENDIF
END RECHERCHE
```

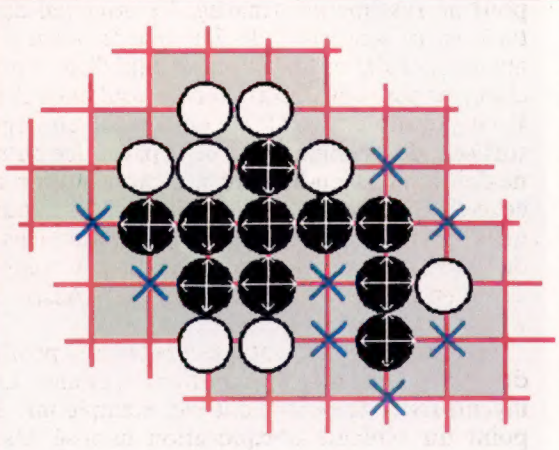
La mise en œuvre de la récursion se fait au moyen de quatre instructions séparées, et non de l'intérieur d'une boucle, afin d'éviter les problèmes qui apparaissent lorsque la recherche est menée à un niveau trop profond.

Une structure en pile

Quand le sous-programme s'appelle lui-même, le paramètre « position de départ » est remplacé par un nouveau (dans l'une des quatre directions), et l'exécution du module continue.

Ce nouveau paramètre peut lui-même provoquer l'appel récursif de la procédure de recherche, créant une nouvelle variable « position de départ », et ainsi de suite. A un moment ou à un autre, une des conditions IF se révélera fausse, aussi l'appel prendra fin sans que les instructions THEN provoquent à leur tour une récurrence. Dans ce cas, le paramètre « position de départ » reprend sa valeur précédente.

C'est donc une structure de données « en pile », qu'on peut comparer à un groupe d'assiettes empilées les unes sur les autres. A chaque appel récursif de la procédure, une nouvelle assiette est placée au sommet de la pile. Quand ce niveau particulier de récurrence prend fin,



Compte en tête

Le programme de recherche, utilisé pour évaluer l'état des groupes du plateau, incorpore (ou simule) des procédures récursives de recherche dans quatre directions à partir de chaque pierre d'un groupe. Le programme s'arrête quand une ouverture ou une pierre ennemie est atteinte.

l'assiette est ôtée de la pile (et la même valeur de la variable est remise en circulation). C'est ce qu'on appelle le principe LIFO (*Last In, First Out,* « dernier entré, premier sorti »).

Le Commodore 64, le Spectrum et l'Amstrad ne disposent pas de procédures et de variables locales. C'est pourquoi, dans les versions qui leur sont destinées, nous emploierons les tableaux SK%() ou s(), qui géreront les valeurs nécessaires.

Pour cela un pointeur (STACK%, ou un équivalent) aura pour tâche de conserver trace de la position placée au sommet de la pile. Il sera incrémenté à chaque nouvel ajout, et décrémenté à chaque retrait. La variable appropriée recevra alors la valeur correspondante — celle du sommet.

Une des conséquences de ce sous-programme est que la récurrence prend fin aux bords du groupe analysé. Quand nous en arrivons là, nous pouvons chercher à voir s'ils se trouvent eux-mêmes aux bords du plateau, ou s'ils sont occupés par des pierres de couleur opposée. Aussi la seconde partie de la procédure fait-elle le décompte des libertés du groupe, à l'aide de la variable CLIB%. Là encore, les positions correspondantes doivent être décomptées, faute de quoi elles seraient incluses dans le total à deux reprises.

Pour le décompte des pierres du groupe, nous ferons appel à la procédure PROCcomptage, qui elle-même appelle PROCrecherche. Cela provoquera l'initialisation des variables de comptage et la mise à zéro des marqueurs après emploi de PROCrecherche.

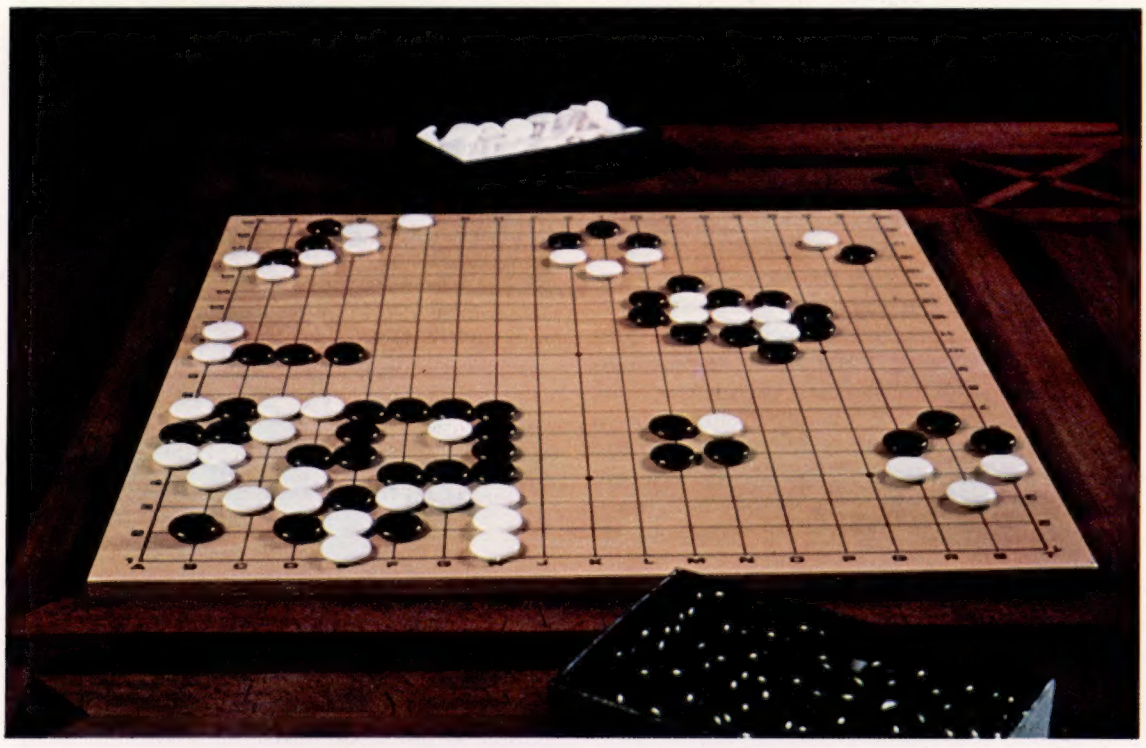
Pour le marquage des positions, vous remarquerez que nous nous servons du plateau de jeu lui-même, afin d'éviter de compter deux fois. Une fois ce plateau marqué, toutefois, il nous faudra nous débarrasser des marqueurs avant de pouvoir continuer. De nouveau nous ferons appel à un sous-programme d'emploi général, PROCmiseàzéro, qui procède à un AND logique (AND 3) sur les octets concernés du plateau, ce qui supprime les marqueurs du bit 2. Les couleurs sont gérées par les deux bits de poids faible de chaque octet, et les marqueurs (pierre, libertés) par les troisième et quatrième bits; PROCmiseàzéro(3) est donc parfaitement adaptée à la tâche. Par exemple :

Octet	A	B	C	D	E	F	G	H
Masque	0	0	0	0	0	0	1	1
Résultat	0	0	0	0	0	0	G	H

La version destinée au Spectrum est un peu différente; la commande AND du BASIC de l'appareil ne laisse pas à découvert les bits qui ne nous intéressent pas, et ne peut être mise en œuvre que pour donner des résultats de type vrai/faux dans des expressions du type IF X < 3 AND Y = 3 THEN... Il faudra donc procéder dans ce cas particulier à une opération de soustraction afin de remettre à zéro le bit 2.

La définition de la procédure PROCmiseàzéro permet par ailleurs de la mettre en œuvre au début de chaque partie, afin de vider complètement le plateau de tout ce qui pourrait s'y trouver : on l'appelle tout simplement avec un paramètre égal à zéro. C'est le sens de la ligne 1400.

Il ne reste qu'une seule procédure à mettre en œuvre avant de pouvoir rédiger tous les modules spécifiquement consacrés aux mouvements des pièces et à leur analyse. C'est celle qui, au tout début du jeu, place sur le plateau les pierres de handicap. Le plus faible des deux joueurs (il joue toujours avec les noirs) a le droit de disposer préalablement sur le plateau entre deux et neuf pierres. Elles doivent impérativement être placées en des points bien précis, définis par le module handicap (lignes 600 à 750). L'ajout des lignes 1580 et 1630 à 1690 permet de mettre en place, de



En raison de son haut degré d'abstraction, le jeu de go a conquis un très large public en Europe. En outre, ses structures de jeu se prêtent admirablement à des exercices de programmation particulièrement intéressants. (Cl. Pix-J. Bénazet.)

façon correcte, les pierres de handicap ; les DATA des lignes 670 à 740 contiennent les positions correspondantes. Vous constaterez que les lignes de DATA sont de longueur croissante ; on passe en effet de deux à neuf pierres.

Le programme a délibérément été conçu sous une forme très structurée, ce qui permet de le modifier aisément afin d'y incorporer une option deux joueurs ; l'ordinateur se borne dans ce cas à vérifier qu'aucun des concurrents ne triche.

Commodore 64

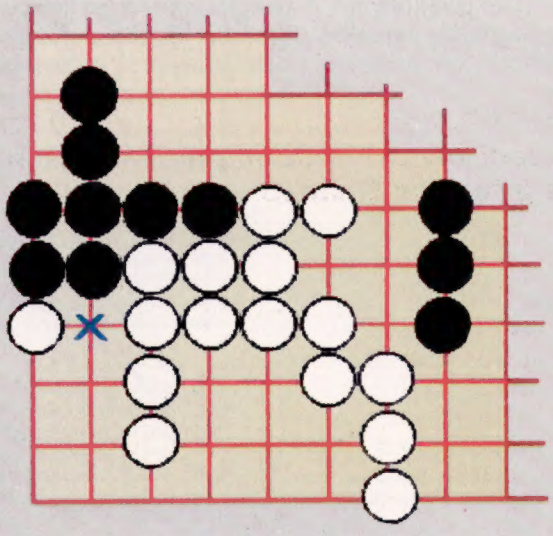
```
320 GOSUB 600
590 :
600 REM READ-HANDICAPS ROUTINE
620 HNCP=BOARD+512
640 FOR L=0 TO 43
650 READ H%:POKE HNCP+L,H%
660 NEXT
670 DATA  68,204
680 DATA  68,204, 76
690 DATA  68,204, 76,196
700 DATA  68,204, 76,196,136
710 DATA  68,204, 76,196,132,140
720 DATA  68,204, 76,196,132,140,136
730 DATA  68,204, 76,196,132,140, 72,200
740 DATA  68,204, 76,196,132,140, 72,200
,136
750 RETURN
760 :
770 REM**********************************
1400 MSK%=0:GOSUB 4330
1580 GOSUB 1630
1620 :
1630 REM HANDICAP ROUTINE
1650 Q%=INT((HND%-2)/2*(HND%+1)+0.5)
1660 FOR L=Q% TO Q%+HND%-1
1670 P%=PEEK(HNCP+L):POKE BOARD+P%,BLACK%
1680 NEXT
1690 RETURN
1700 :
1710 REM**********************************
4030 :
4040 REM COUNT ROUTINE
4050 CLIB%=0:CSTN%=0
4070 SP%=CP%:SC%=CC%:GOSUB 4130
4080 MSK%=COLOUR%:GOSUB 4330
4090 RETURN 0 THEN RETURN
4150 IF (PEEK(BOARD+SP%) AND COLOUR%)=0
GOTO 4250
4160 IF (PEEK(BOARD+SP%) AND SC%)=0 THEN
 RETURN
4170 IF (PEEK(BOARD+SP%) AND MARKER%)>0
THEN RETURN
4100 :
4110 REM**********************************
4120 :
4130 REM SEARCH ROUTINE
4140 IF (SP% AND 240)=0 OR (SP% AND 15)=
4180 POKE BOARD+SP%,SC%+MARKER%
4190 CSTN%=CSTN%+1
4195 SK%(STACK%)=SP%:STACK%=STACK%+1
4200 SP%=SK%(STACK%-1)+DIR%(1):GOSUB 413
0
4210 SP%=SK%(STACK%-1)+DIR%(2):GOSUB 413
0
4220 SP%=SK%(STACK%-1)+DIR%(3):GOSUB 413
0
4230 SP%=SK%(STACK%-1)+DIR%(4):GOSUB 413
0
4235 STACK%=STACK%-1:SP%=SK%(STACK%)
4240 RETURN
4250 IF (PEEK(BOARD+SP%) AND LIBERTY%)>0
 THEN RETURN
4260 POKE BOARD+SP%,LIBERTY%
4270 CLIB%=CLIB%+1
4290 RETURN
4300 :
4310 REM**********************************
4320 :
4330 REM CLEAR ROUTINE
4350 FOR L=0 TO 255
4360 POKE BOARD+L,PEEK(BOARD+L) AND MSK%
4370 NEXT
4380 RETURN
4390 :
4400 REM**********************************
```

Amstrad CPC 464/664

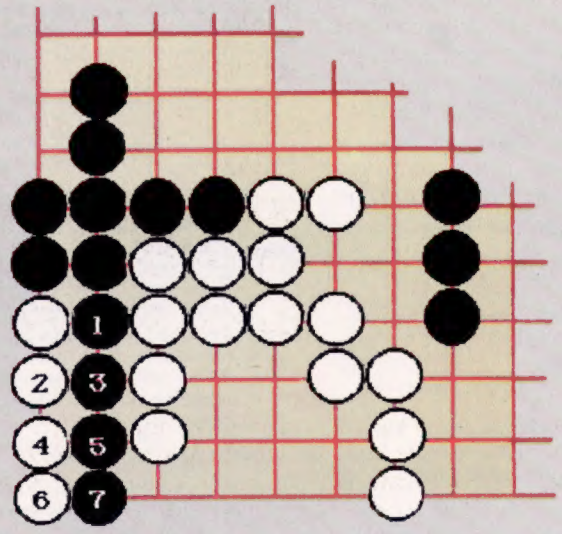
```
270 GOSUB 600:REM read handicaps
590 :
600 REM read handicaps routine
620 hncp=board+&200
630 RESTORE 670
640 FOR i%=0 TO 43
650 READ h%:POKE(hncp+i%),h%
660 NEXT i%
670 DATA &44,&cc
680 DATA &44,&cc,&4c
690 DATA &44,&cc,&4c,&c4
700 DATA &44,&cc,&4c,&c4,&88
710 DATA &44,&cc,&4c,&c4,&84,&8c
720 DATA &44,&cc,&4c,&c4,&84,&8c,&88
730 DATA &44,&cc,&4c,&c4,&84,&8c,&48,&c8
740 DATA &44,&cc,&4c,&c4,&84,&8c,&48,&c8
,&88
750 RETURN
760 :
770 REM **********************
1400 mask%=0:GOSUB 4330:REM clear routine
1580 GOSUB 1630:REM hahdicap routine
1620 :
1630 REM handicap routine
1650 q%=INT((hand%-2)/2*(hand%+1)+0.5)
1660 FOR i%=q% TO q%+hand%-1
1670 LET p%=PEEK(hncp+i%):POKE(board+p%)
,black%
1680 NEXT i%
1690 RETURN
1700 :
1710 REM **********************
4030 :
4040 REM count routine
4050 clib%=0:cstn%=0
4060 cloc%(1)=0:cloc%(2)=0
4070 sp%=cp%:sc%=cc%:GOSUB 4130:REM search
4080 mask%=colour%:GOSUB 4330:REM clear
4090 RETURN
4110 REM **********************
4120 :
4130 REM search routine
4140 IF (sp% AND 240)=0 OR (sp% AND 15)=
0 THEN RETURN
4150 IF (PEEK(board+sp%)AND colour%)=0 T
HEN 4250
4160 IF (PEEK(board+sp%)AND sc%)=0 THEN
RETURN
4170 IF (PEEK(board+sp%)AND marker%)>0 T
HEN RETURN
4180 POKE(board+sp%),sc%+marker%
4190 cstn%=cstn%+1
4195 s(stack%)=sp%:stack%=stack%+1
4200 sp%=s(stack%-1)+dir%(1):GOSUB 4130
4210 sp%=s(stack%-1)+dir%(2):GOSUB 4130
4220 sp%=s(stack%-1)+dir%(3):GOSUB 4130
4230 sp%=s(stack%-1)+dir%(4):GOSUB 4130
4235 stack%=stack%-1:sp%=s(stack%)
4240 RETURN
4250 IF (PEEK(board+sp%) AND liberty%)>0
 THEN RETURN
4260 POKE(board+sp%),liberty%
4270 clib%=clib%+1
4290 RETURN
4300 :
4310 REM **********************
4320 :
4330 REM clear routine
4350 FOR i%=0 TO 255
4360 POKE(board+i%),(PEEK(board+i%) AND
mask%)
4370 NEXT i%
4380 RETURN
4390 :
4400 REM **********************
```

Sinclair Spectrum

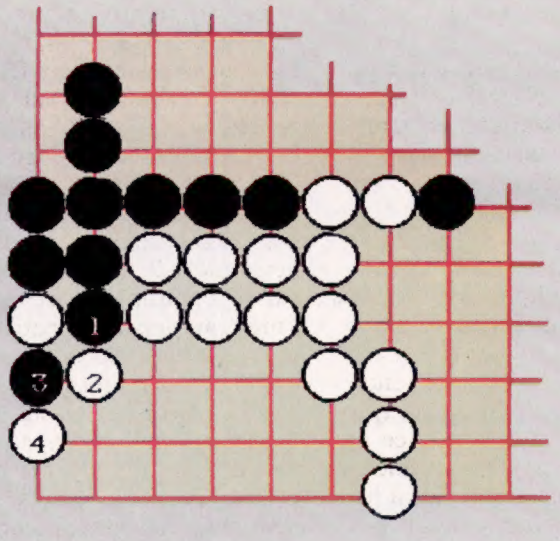
```
270 GO SUB 600
590:
600 REM read-handicaps routine
620 LET hncp=board+512
630 RESTORE 670
640 FOR l=0 TO 43
650 READ h: POKE hncp+l,h
660 NEXT l
670 DATA  68,204
680 DATA  68,204, 76
690 DATA  68,204, 76,196
700 DATA  68,204, 76,196,136
710 DATA  68,204, 76,196,132,140
720 DATA  68,204, 76,196,132,14
0,136
730 DATA  68,204, 76,196,132,14
0, 72,200
740 DATA  68,204, 76,196,132,14
0, 72,200,136
750 RETURN
760:
770 REM ************************
1400 LET mask=0: GO SUB 4330
1580 GO SUB 1630
1620:
1630 REM handicap routine
1650 LET q=INT ((hand-2)/2*(hand
+1)+0.5)
1660 FOR l=q TO q+hand-1
1670 LET p=PEEK (hncp+l): POKE b
oard+p,black
1680 NEXT l
1690 RETURN
1700:
1710 REM ************************
4030:
4040 REM count routine
4050 LET clib=0: LET cstn=0
4070 LET sp=cp: LET sc=cc: GO SU
B 4130
4080 LET mask=colour: GO SUB 4330
4090 RETURN
4100:
4110 REM ************************
4120:
4130 REM search routine
4140 IF INT (sp/16)=0 OR sp-16*I
NT (sp/16)=0 THEN  RETURN
4150 IF PEEK (board+sp)=liberty
OR PEEK (board+sp)=0 THEN  GO TO
 4250
4160 IF PEEK (board+sp)=colour-s
c THEN  RETURN
4170 IF PEEK (board+sp)>colour T
HEN  RETURN
4180 POKE board+sp,sc+marker
4190 LET cstn=cstn+1
4195 LET s(stack)=sp: LET stack=
stack+1
4200 LET sp=s(stack-1)+d(1): GO
SUB 4130
4230 LET sp=s(stack-1)+d(4): GO
SUB 4130
4235 LET stack=stack-1: LET sp=s
(stack)
4240 RETURN
4250 IF PEEK (board+sp)>colour T
HEN  RETURN
4260 POKE board+sp,liberty
4270 LET clib=clib+1
4290 RETURN
4300:
4310 REM ************************
4320:
4330 REM clear routine
4350 FOR l=0 TO 255
4360 IF PEEK (board+l)>mask THEN
  POKE (board+l),PEEK (board+l)-
mask-1: GO TO 4360
4370 NEXT l
4380 RETURN
4390:
4400 REM ************************
```



Avantage territorial
Ici, les blancs se sont emparés du coin inférieur gauche du plateau, mais leur territoire n'est pas encore véritablement sûr. Le point vital en ce sens est marqué d'une croix.



Course de vitesse
Si en effet les noirs ont l'occasion de jouer à cet endroit, ce sera un *atari* sur la pierre blanche située tout au bord à gauche. Les blancs peuvent bien sûr essayer d'occuper le coin, mais ils ne pourront éviter d'être finalement capturés par leur adversaire.



Une solution de rechange
Plutôt que d'occuper le coin, les blancs peuvent choisir de sacrifier leur pierre, et par là même de contenir l'invasion par les noirs de leur propre territoire.

Caroline Clayton sur Macintosh

Vision particulière

L'Omni-reader, d'Oberon International, est l'un des derniers OCR à microprocesseur pouvant lire un texte imprimé et le charger dans un micro possédant une interface RS232C.

Au cours des dernières années, on a beaucoup parlé de l'évolution du travail de bureau et de la disparition progressive de la paperasse. Puisque les ordinateurs peuvent échanger de l'information électroniquement, pourquoi s'encombrer de papier? Cependant, les scénarios que certains avaient prématurément envisagés se sont révélés inexacts. La plupart des utilisateurs préfèrent encore utiliser le papier pour certains traitements de l'information. De plus, les données doivent encore être entrées manuellement dans l'ordinateur en attendant le travail de bureau purement électronique. Pour accélérer ce mouvement, les unités de reconnaissance optique de caractères, qui firent leur apparition en 1955, lisent le texte sur une page imprimée et l'entrent dans l'ordinateur. Cependant, ce n'est que tout récemment qu'elles ont commencé à faire leur apparition dans les bureaux.

L'Omni-reader d'Oberon International est l'une de ces nouvelles unités de gestion. Bien que le logiciel ne s'adresse initialement qu'à l'IBM PC, à la gamme Apricot et à l'Apple Macintosh, l'unité peut théoriquement être utilisée par tout micro possédant une interface RS232C.

L'Omni-reader est composé d'une tablette porte-document en plastique et d'une règle horizontale qui glisse le long d'une barre verticale du côté gauche de la tablette. Le lecteur optique lui-même (un dispositif manuel muni de deux boutons et d'une DEL) est monté sur la règle et peut être déplacé sur la surface d'un document placé sur la tablette, un peu comme la tête d'impression d'un traceur xy.

Différentes fonctions et polices de caractères peuvent être sélectionnées; celles-ci sont représentées par des DEL qui apparaissent en haut de la tablette. Quatre polices seulement peuvent être analysées par l'Omni-reader. Ce sont Courier 10, Courier 12, Gothique 12 et Prestige Elite 12, lesquelles sont les plus utilisées sur les imprimantes à marguerite et sur les machines à écrire.

A l'intérieur du lecteur optique, deux sources de lumière infrarouge dirigent l'image sur les circuits détecteurs de lumière. Le détecteur ne peut lire que les caractères imprimés avec des encres au carbone ou résultant d'une photocopie (procédé également fondé sur le carbone), ce qui implique des avantages et des inconvénients. L'aspect positif est que l'encre des stylos à bille et les autres encres sont « transparentes » pour le détecteur. Le texte peut donc être lu même s'il comporte des marques de stylo ou des coups de tampon. De plus, cela signifie que la plupart des papiers colorés ne gêneront pas le fonctionnement de l'Omni-reader. En revanche, les limites du système sont évidentes quand on imagine que les traits de crayon doivent être entièrement gommés afin qu'un texte soit lu correctement.

Un peu de lecture...
L'Omni-reader d'Oberon est la première unité de reconnaissance de caractères (OCR) qui soit à la fois assez fiable et relativement peu coûteuse. Le texte d'une copie imprimée peut être lu par l'Omni-reader et transféré directement dans la mémoire de l'ordinateur sans avoir à taper le document. Actuellement, Oberon n'a produit que les logiciels correspondant aux ordinateurs de gestion les plus répandus, bien que tout ordinateur doté d'une interface RS232C puisse utiliser le système.
(Cl. Crispin Thomas.)

Le lecteur optique est relié à l'arrière de la tablette de l'Omni-reader au moyen d'un câble. Apparaissent également à l'arrière de la tablette une prise d'alimentation externe, une prise RS232C de type D standard à 25 broches et deux blocs de commutateurs DIP — l'un sert à régler le débit en bauds et l'autre permet de définir des fonctions comme l'établissement de liaison et l'espacement de caractères.

Pour lire du texte dans l'ordinateur, vous devez d'abord placer la copie imprimée sur la tablette, puis aligner une fenêtre définie par la règle au-dessus de la ligne devant être lue. Il faut appuyer sur le bouton du lecteur optique et déplacer l'unité au-dessus du papier; elle lit ainsi ce qui y est inscrit. Après environ une seconde, si l'Omni-reader a réussi à lire la ligne, la DEL clignote une fois, et un signal sonore est produit; finalement la ligne apparaît à l'écran. Si le lecteur n'a pas lu la ligne correctement, il clignote et produit deux signaux sonores. L'utilisateur peut alors choisir de recommencer la lecture de la ligne au moyen d'un autre passage, ou demander à l'ordinateur d'accepter la ligne en appuyant sur le bouton du haut.

L'Omni-reader compare la forme qu'il lit à la liste des caractères qu'il possède pour ce type particulier de police. Les caractères sont contenus dans des modèles de bits mappés dans la mémoire de l'Omni-reader, et l'unité trouve quels sont les caractères en examinant constamment l'image apparaissant dans l'objectif du lecteur. L'image est ensuite divisée en cinquante « tranches » dans les directions horizontale et verticale.

Omni-reader

INTERFACES

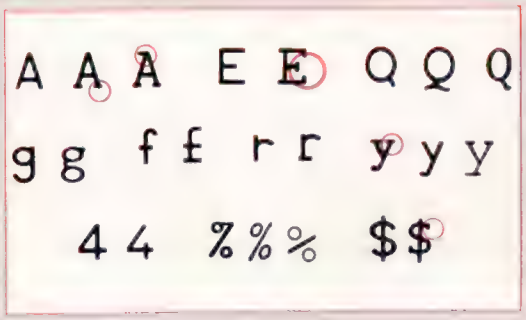
L'interface standard RS232C avec débit en bauds ajustable.

FORCES

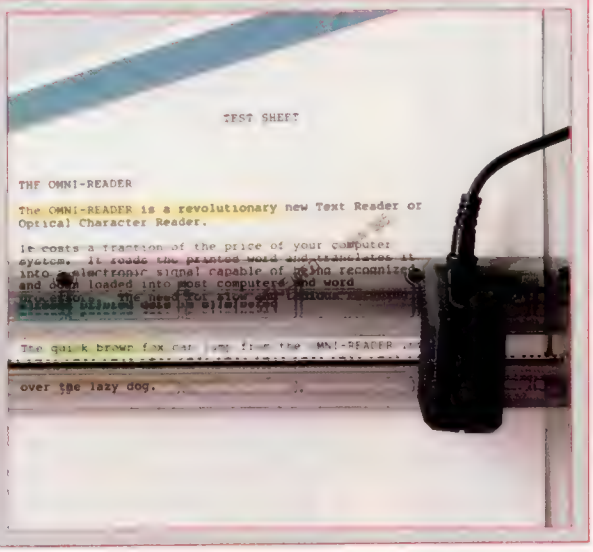
L'Omni-reader est rapide et précis et peut entrer des copies dans un ordinateur beaucoup plus rapidement qu'en les tapant au clavier.

FAIBLESSES

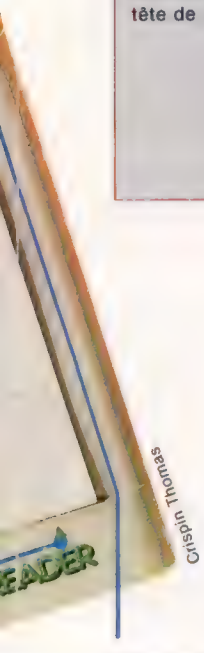
Diverses contraintes matérielles limitent la lecture à certaines polices de caractères. Puisque l'unité ne peut lire que des encres au carbone, cela limite encore un peu plus son domaine d'utilisation.



Caractères ambigus
Bien que les polices de caractères semblent assez similaires, il existe certaines différences qui compliquent leur détection électronique. La principale difficulté provient des différences aux extrémités (encerclées) des lettres. Puisque l'Omni-reader utilise des modèles prédéterminés en mémoire, toute lettre qui ne correspond pas exactement à ces modèles sur la page imprimée est ignorée. Cependant, même si certaines ressemblances apparaissent, des lettres comme *Q* et *g* sont vraiment différentes.



Une fenêtre
Afin que l'unité de reconnaissance optique puisse fonctionner correctement, le texte doit être positionné exactement au centre de la fenêtre. Au bas de la fenêtre d'examen existe une série de blocs, similaires à des codes à barres. Lors de l'examen, ceux-ci indiquent à l'Omni-reader dans quelle direction se déplace la tête de lecture.



Guide visuel
Le lecteur optique de caractères est guidé manuellement le long de la ligne de texte. A l'extrémité d'une ligne, la DEL produit un seul clignotement. Un clignotement signifie une lecture réussie tandis que deux clignotements indiquent un échec partiel.

Lors de son déplacement latéral, l'Omni-reader recherche des caractères entiers sous sa tête de lecture. Il ne traite que les caractères situés au centre de son champ de vision et qui ne sont pas fractionnés par l'une des bordures de la zone examinée. Le caractère est ensuite envoyé dans la mémoire de l'ordinateur afin de lui trouver une correspondance.

Quand l'Omni-reader trouve une correspondance, le logiciel envoie le code ASCII équivalent *via* l'interface RS232C et le caractère apparaît à l'écran. Puisque l'Omni-reader doit être en mesure de trouver une correspondance exacte dans les modèles contenus en mémoire, le texte doit être correctement aligné à l'intérieur de la fenêtre d'examen. Par conséquent, si les jambages (le bas des lettres *y, g, p* et *q*) sont situés à l'extérieur de la fenêtre, ils peuvent créer une erreur — par exemple, un *p* peut être interprété comme étant un *o*.

Aussi, si la fenêtre est trop basse, le lecteur optique peut lire le haut des caractères de la ligne suivante et essayer de les interpréter. Des problèmes peuvent également se poser lorsque l'utilisateur essaie d'entrer un texte trop grand ou trop petit pour entrer dans la zone à examiner. Oberon recommande de n'utiliser que quatre ou cinq caractères par centimètre. Tout ce qui se situe hors des limites peut générer une erreur.

Malgré ces contraintes, l'Omni-reader fonctionne étonnamment bien. Le meilleur temps de lecture d'une ligne A4 est compris entre 0,5 et 1,5 seconde, ce qui équivaut au temps nécessaire pour glisser l'unité d'un côté à l'autre de la page. Bien que le débutant ait au départ quelques difficultés à aligner le texte correctement, il réussit généralement assez rapidement à bien estimer l'emplacement ; en moins d'une heure d'utilisation, il est possible de produire des lignes sans difficulté et sans erreur.

Options et commandes

Afin de faciliter la lecture de polices de caractères de mauvaise qualité, certaines options ont été ajoutées. Ces options sont fournies à l'Omni-reader en passant le lecteur optique au-dessus de commandes écrites — quatre sont placées au-dessus de la tablette et les autres sont décrites dans le manuel. Chaque commande doit être précédée de deux carrés afin qu'elle ne soit pas interprétée comme un mot devant être affiché à l'écran.

Parmi les quatre commandes qui sont définies sur la tablette, la plus utilisée est TYPEFACE, qui vous permet de choisir la police devant être lue. NUMERIC retire les caractères alphabétiques de la police et ne lit que les caractères numériques, ce qui accélère la lecture. Cette option est probablement l'application la plus utile de l'Omni-reader puisque la saisie de chiffres est généralement l'opération qui génère le plus d'erreurs non détectées au départ.

L'entrée directe est une bonne méthode pour éviter ce problème. Pour traiter des imprimés de mauvaise qualité, l'option POOR peut être utilisée. Elle ralentit le rythme de lecture de l'Omni-reader afin qu'il puisse obtenir une image nette du caractère situé sous le lecteur optique. Elle est particulièrement utile lors de la lecture d'un document réalisé à partir d'un ruban en tissu assez usé.

Le logiciel Omni-reader consiste en deux modules de programmes qui résident dans la zone système de la mémoire. Cela signifie que les programmes d'application peuvent être chargés dans la mémoire en même temps. Les données d'un document imprimé peuvent donc être entrées directement dans un programme de traitement de texte comme WordStar ou, si vous lisez des chiffres, peuvent être directement insérées dans les colonnes d'un tableur afin de les calculer instantanément.

L'Omni-reader coûte plus de 10 000 F, logiciel et câble de connexion compris. Le fabricant propose également une demi-journée de formation et un entretien pendant un an. Ce n'est vraiment pas donné. Cependant, pour une entreprise qui reçoit de nombreux documents imprimés, comme une entreprise de presse, ce type d'investissement peut se révéler très rentable.

Petite tortue

Vous connaissez le langage pédagogique logo. Voyons Dr Logo pour ordinateurs Amstrad, et comparons-le avec l'original destiné à l'IBM PC.

Quand on a voulu transférer LOGO des gros systèmes vers les micro-ordinateurs, il dut subir un certain nombre de restrictions et de simplifications pour être accepté par les 64 K (ou moins) de mémoire disponible sur ces machines. Lorsque des micros 16 bits firent leur apparition avec plus de mémoire, des nouvelles versions améliorées arrivèrent rapidement sur le marché.

Gary Kildall — fondateur de Digital Research et du système d'exploitation CP/M — fut conquis par les potentialités du langage LOGO, et produisit une version appelée Dr LOGO. Cette version a depuis été mise en œuvre sur un certain nombre d'autres machines 16 bits. L'Apricot F1e, par exemple, est livré avec Dr LOGO.

Digital Research a depuis adapté Dr LOGO à un certain nombre de machines CP/M 80 sur 8 bits. Amstrad propose une de ces versions avec son kit logiciel accompagnant le CPC 464.

Dr LOGO, tel qu'il existe sur IBM PC, suppose au moins 192 K. Le disque est protégé en copie, mais une copie de sauvegarde est fournie. Pour exploiter Dr LOGO, il suffit d'introduire le disque et de faire une réinitialisation système. La résolution graphique couleur est de 320 × 200 pixels, avec un choix de seize couleurs de fond et de quatre jeux de trois couleurs pour le tracé.

Avec une carte graphique sur l'IBM PC, et à la fois un écran couleur et un moniteur monochrome, Dr LOGO peut produire et du graphique sur l'écran couleur, et du texte sur le moniteur monochrome ! Si vous ne disposez cependant que d'un moniteur monochrome, les deux modes peuvent être mixés sur le même écran.

Dr LOGO est décrit dans le manuel comme un sur-ensemble du langage LOGO, et contient de fait toutes les commandes LOGO : les commandes standards graphiques, le traitement de listes et de gestion d'espace de travail, les primitives de gestion d'erreurs. Mais aussi les assignations multiples sur un seul objet et les ensembles (groupage de procédures). Dr LOGO oblige à entrer les commandes en minuscules.

L'éditeur utilise les touches de fonction IBM mais reconnaît aussi les codes de contrôle du LOGO Apple. Quand une erreur est trouvée à l'exécution d'un programme, il suffit de taper ed pour transmettre à l'éditeur la procédure en cause, prête à être corrigée. Il n'existe malheureusement pas de moyen de savoir, en regardant l'écran, si l'on est en mode édition. La plupart des autres versions l'indiquent par une ligne en bas de l'écran.

Banc d'essai logo

Procédure		Résultats			
		IBM PC (256 K)	Amstrad CPC 464/664	Commodore 64	Apricot F1e
RECUR	Niveau atteint	512	187	90	932
DEFVARS	Niveau atteint	1307	563	553	1806
RECURFIN1	Niveau atteint	512	188	infini	infini
RECURFIN2	Niveau atteint	151	129	86	72
LIGNES (avec tortue)	Chrono (à la seconde près)	9	53	13	10
LIGNES (sans tortue)	Chrono (à la seconde près)	5	27	11	6
ASSIGNER	Chrono (à la seconde près)	25	22	13	20
LISTES	Chrono (à la seconde près)	64	74	18	69
ARITH	Chrono (à la seconde près)	80	86	38	77
RECURA	Chrono (à la seconde près)	92	91	24	72

Les procédures banc d'essai

Les neuf procédures du banc d'essai LOGO sont conçues pour vérifier l'accomplissement de diverses fonctions et pour montrer les points forts et les inconvénients de chaque version.

- RECUR évalue la taille de la pile.
- DEFVARS calcule la place disponible pour la définition des variables et des procédures.
- RECURFIN1 et RECURFIN2 déterminent l'efficacité de l'application de la récursion finale. L'idéale devrait permettre une récurrence à l'infini. Alors que RECURFIN2 donne un résultat et peut être considérée comme une opération, RECURFIN1 est une commande.
- LIGNES mesure la vitesse de tracé. Deux chiffres permettent de mettre en évidence le gain de temps obtenu en cachant la tortue.
- ASSIGNER évalue la vitesse à laquelle des variables reçoivent des valeurs.
- LISTES apprécie la vitesse des opérateurs de traitement de listes.
- ARITH mesure la vitesse d'exécution des fonctions arithmétiques.
- RECURA calcule la vitesse des appels récursifs.

En dehors des primitives habituelles de traitement de listes, Dr LOGO propose :

— TRI pour le tri d'une liste par ordre alphabétique;

— BATTRE pour battre les éléments comme on bat des cartes, afin de les disposer dans un ordre quelconque;

— PARTAGER vous permet de ne retenir qu'une partie de la liste.

Dr LOGO ajoute également un certain nombre d'outils de dépistage d'erreurs :

— SURVEILLANCE pour suivre ligne par ligne les effets d'une procédure;

— TRACER affiche les noms des variables au fur et à mesure qu'ils sont appelés et définis;

— DÉPISTAGE D'ERREURS divise l'écran en deux fenêtres : la fenêtre de dépistage affiche les informations de SURVEILLANCE et de TRACER, la « fenêtre de programme » donne le résultat du programme.

On peut ajouter des commentaires aux procédures, et on peut aussi les supprimer de l'espace de travail par SANSFORMAT, si l'on a besoin de davantage de place pour l'exécution du programme.

Il existe également un certain nombre de nouvelles primitives de gestion :

— ORDRE vous permet de définir l'ordre dans lequel se présentent les primitives à l'écran;

— POTEL donne le nom des procédures non appelées par d'autres procédures;

— POREF <NOM> affiche le nom des procédures qui appellent la procédure <NOM>;

— POCALL <NOM> affiche le nom des procédures appelées par la procédure <NOM>.

Ces caractéristiques supplémentaires apportent un développement plus étendu aux programmes LOGO.

Des points faibles

Dr LOGO comporte aussi un certain nombre de points faibles : l'éditeur n'a pas de fonctions de recherche et de remplacement, pas de méthode d'interfaçage code machine, et pas de gestion de fichier. Le manuel de trois cents pages est néanmoins très clair et complet. Il comporte une introduction à LOGO et un manuel de référence avec une page par primitive.

Manifestement, Dr LOGO apporte un « plus » par rapport aux autres versions LOGO, ce qui se traduit par un environnement plus riche.

Il est intéressant de comparer Dr LOGO avec les autres versions 8 bits en termes d'espace mémoire et de vitesse d'exécution. L'espace de travail de LOGO se mesure en « nœuds ». Si l'on demande à Dr LOGO de combien de nœuds il dispose au démarrage du système, la réponse est de 10 000 (contre 2 000 à 3 000 pour les autres systèmes 8 bits). Il y a donc largement la place pour définir les procédures et exécuter les procédures récursives. Néanmoins, la récurrence finale — quand une procédure récursive figure à la dernière ligne — n'est pas mise en œuvre efficacement. Aussi, les programmes qui fonctionneraient indéfiniment sur Commodore, par exemple, sont rapidement à cours de mémoire.

Pour ce qui concerne la vitesse, le mode graphique est rapide, mais le calcul arithmétique et le traitement de liste sont plus lents qu'avec toute autre version 8 bits du langage.

Version Amstrad

Dr LOGO est livré avec le lecteur Amstrad et figure sur l'autre face du disque CP/M. Dr LOGO est modifié pour Amstrad afin de tenir compte du système. Le graphique utilise les mêmes quatre couleurs de tracé que l'IBM PC, mais elles sont définies par une liste de nombres qui déterminent les proportions de rouge, de vert et de bleu. La version IBM ne dispose pas d'une gamme très étendue de commandes sonores, alors que la version Amstrad possède env (enveloppe sonore), ent (enveloppe tonale) et release, qui libère les canaux affectés à un état continu dans une commande sonore.

La version Amstrad comporte la plupart des caractéristiques du LOGO Apple, dont l'assignation multiple à un même objet et la gestion d'erreurs, mais pas d'utilitaires.

Les primitives de dépistage d'erreurs et les autres caractéristiques de gestion de procédures de Dr LOGO sur IBM PC ont été laissées de côté sur Amstrad, probablement à cause de ses moindres capacités mémoire. Plus ennuyeux, cependant, est le manque de fonction définition. De même, l'absence de moyen pour supprimer un fichier, en dehors de quitter et de réintégrer CP/M, est très préjudiciable.

L'éditeur est très lent à répondre, et il se peut que vous vous trouviez à un moment donné en train de taper des caractères excédentaires qui ne seront pas pris en compte. Mais, en revanche, si une erreur est détectée dans une procédure en cours d'exécution, l'éditeur positionne le curseur sur le mot incriminé, ce qui est très utile.

La version Amstrad permet un accès limité au code machine *via* examine et deposite (PEEK et POKE), ce que ne fait pas l'IBM PC.

La documentation fournie est minimale, et on vous demande de vous référer à un guide utilisateur et à un guide de références déjà publiés. En attendant, vous devrez vous contenter du sommaire des commandes sur vingt-cinq pages. Il semble aussi qu'il y ait quelques caractéristiques de Dr LOGO déjà en place mais non décrites.

De nombreuses fonctions très attrayantes de Dr LOGO sur IBM PC ont disparu sur Amstrad. Il nous reste une bonne version standard LOGO, avec quelques extra tels que les groupements de procédures et la gestion d'erreurs.

Dr LOGO est supposé disposer de 2 105 nœuds au démarrage, ce qui correspond à un chiffre moyen pour un système 8 bits. Cette capacité se trouve néanmoins très fortement compromise par le fait que la récurrence finale n'a pas été efficacement appliquée. En outre, le graphisme, le traitement de listes et les calculs arithmétiques sont très lents. L'ensemble fait qu'il s'agit de l'une des versions 8 bits les plus laborieuses du langage LOGO; la plupart des opérations prennent environ trois fois plus de temps que sur le C 64.

Drame en série

L'Interface 1 fournit toutes sortes de routines utiles en langage machine Spectrum. Nous allons voir les codes crochets dont elle se sert pour contrôler le port série RS232.

Commençons par considérer l'interface série, qui est une prise RS232 270° à neuf broches (la figure montre la fonction de chaque broche). Le nombre de lignes que vous allez avoir besoin de connecter dépend de la complexité de votre application, et une liaison minimale RS232 peut être établie à l'aide des broches 2, 3 et 7. Dans ce cas, les broches 4 et 5 doivent être connectées à la broche 9 pour les maintenir à la logique 1. Cela évite des incidents si quoi que ce soit se trouvant à l'autre extrémité de la liaison n'est pas encore prêt à recevoir des données de l'ordinateur. Toutefois, cela a l'inconvénient de perdre des données si l'autre extrémité de la liaison n'est pas prête à recevoir.

Une application possible de la liaison RS232 est de connecter le Spectrum à une « vraie » imprimante. Une fois câblé, le logiciel de contrôle est simple à utiliser. Pour envoyer du texte à l'imprimante, nous fixons un débit en bauds approprié à l'aide de l'instruction BASIC FORMAT, puis exécutons le code suivant :

```
CLOSE #3
OPEN #3, «T»
```

Cela fait correspondre le flux 3, qui est généralement celui de l'imprimante ZX, au canal T, qui est celui du texte sur la liaison RS232. Tous les détails concernant l'utilisation des flux « texte » et « binaire » sont donnés dans le manuel de l'Interface 1. LLIST et LPRINT enverront des données au dispositif connecté à l'Interface 1.

Considérons maintenant les détails de fonctionnement de l'interface série. Les données sont transmises avec un bit de début, huit bits de données et deux bits stop. Nous n'avons pas à nous occuper de ce format, étant donné que le système d'exploitation du Spectrum le fait à notre place. Ce qui est plus important pour nous est le débit auquel les données sont envoyées (en bauds). Celui-ci est généralement fixé par la commande FORMAT, qui nous limite à un certain intervalle de débits en bauds. Bien qu'il y ait peu de perte d'intégrité de données avec une interface RS232 (même sur une distance considérable), il est recommandé d'utiliser des débits de transmission d'autant plus lents que les distances sont grandes entre émetteur et récepteur.

Il est pratique d'introduire des variables système de l'Interface à l'aide du code crochet 49 avant d'utiliser l'une des techniques suivantes. Il y a deux codes crochets concernés par RS232, et trois variables système. Voyons d'abord les variables.

• BAUD (aux emplacements 23747 et 23748) est une variable à deux octets, qui contient une valeur déterminant le débit à utiliser pour la transmission et la réception de données.

Cela nous montre une des limites de l'interface série Spectrum : son incapacité à avoir simultanément différents débits pour l'émission et la réception.

• SERFL (à l'emplacement 23751) est un octet de drapeau, qui est utilisé par le SE uniquement à la réception. S'il est à 1, il indique qu'il y a encore un octet disponible à lire.

• SERBYT (à l'emplacement 23752) agit comme un tampon d'entrée à un octet pour la réception. Il est occasionnellement possible pour un octet d'être reçu et stocké dans cet emplacement après l'interruption de la communication. Des problèmes peuvent survenir en utilisant les deux dernières variables, que nous considérerons ultérieurement.

Débits en bauds utilisant l'Interface 1

Le standard RS232 est très utilisé dans les communications, en particulier pour des transmissions de données sur de longues distances, et le *débit en bauds* fournit une indication du nombre de bits de données transmis par seconde. En appliquant la formule indiquée dans le texte à quelques-uns des débits les plus courants, on obtient les valeurs suivantes :

Débit en bauds	Valeur
50	2 690
110	1 221
300	446
600	222
1 200	110
2 400	54
4 800	28
9 600	12
19 200	5

La valeur du débit désiré est alors POKée dans la variable système BAUD aux adresses 23747 et 23748. La variable système unique implique, malheureusement, qu'il faut utiliser le même débit à la fois pour émettre et recevoir des données.
La partie de code suivante montre comment utiliser ces valeurs. Pour fixer le débit à 300 bauds :

```
LD      HL, 446
LD      (23747), HL
RET
```

Une quatrième variable système (IOBORD [à l'emplacement 23750]) n'est pas strictement concernée par l'interface série, mais elle contient la couleur de la marge d'écran à utiliser dans les opérations E/S. Vous pouvez donc la fixer pour contenir la couleur que vous désirez pour la marge d'écran pendant ces opérations.

Quand nous voulons utiliser l'interface RS232, la première chose à faire est de fixer un débit en bauds. Cela se fait en mettant une valeur appropriée dans BAUD — en fait, c'est tout ce que fait FORMAT.

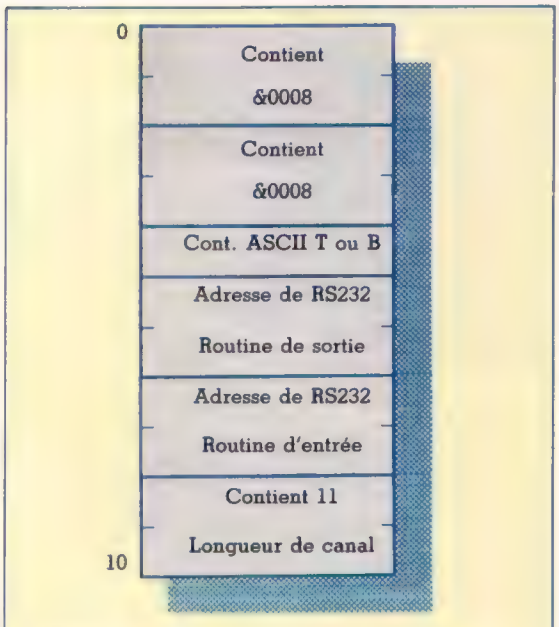
La valeur à utiliser pour un débit particulier est donnée par :

valeur = (350 000/[26 × débit en bauds]) − 2.

En utilisant cette équation, il est possible de fixer pour l'interface série un débit non standard à partir du langage machine, mais c'est probablement d'un intérêt mineur à moins que vous ne communiquiez avec un autre Spectrum *via* RS232. Le tableau des débits en bauds donne l'ensemble des valeurs de débits et la routine nécessaire pour les mettre en œuvre.

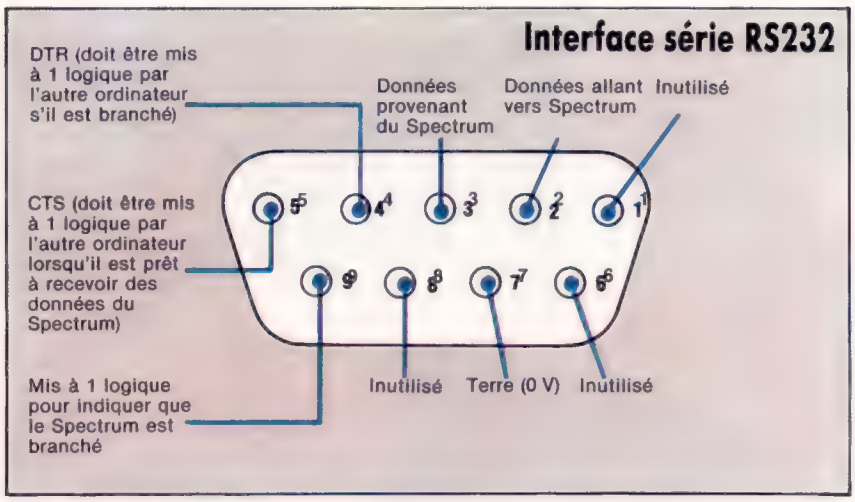
Considérons maintenant la façon dont les données sont transférées à travers la liaison. En BASIC, l'ouverture (OPEN) d'un canal à utiliser avec l'interface RS232 génère un canal de la forme montrée à la figure « Structure de canal RS232 ». Ce canal est placé dans la zone canal de mémoire. L'octet 4 de ce canal voit ajouter 128 à sa valeur si le canal a été créé implicitement (c'est-à-dire à l'aide de SAVE*, MOVE, FORMAT, etc.).

Toutefois, ce n'est pour nous que d'un intérêt académique lorsque nous en venons à utiliser les routines RS232 à partir du langage machine. Vous remarquerez qu'il n'y a pas de tampon associé à un tel canal. Nous lisons et écrivons les données en appelant le code crochet adéquat, qui



Structure de canal RS232
Notez qu'il n'y a pas de tampon associé au canal. Toutes les données à émettre ou à recevoir passent dans le registre A *via* les codes crochets 29 et 30.

Caroline Clayton



à son tour appelle la routine ayant son adresse dans les emplacements 5 et 6 de canal (pour transmission) ou 7 et 8 (pour réception). Quand nous utilisons un code crochet, aucun canal n'est fixé. Les deux codes crochets RS232 sont :

- *Code crochet 29.* Lorsqu'il est appelé, il donne un seul octet depuis l'interface série dans le registre A, s'il y en a un de disponible (auquel cas le drapeau de retenue sera mis à 1). S'il n'y a pas d'octet disponible, alors A contiendra zéro et le drapeau de retenue sera effacé.
- *Code crochet 30.* Il transfère l'octet dans le registre 2 vers l'interface série pour transmission au débit en bauds en cours. Quand nous utilisons ce code, tous les octets sont transmis sans modification; il n'y a pas de différence entre les flux B et T au niveau du langage machine.

C'est à ce point que les problèmes concernant SERBYT et SERFL peuvent se manifester. Si la routine code crochet trouve que SERFL est mis à 1, alors il reçoit un octet de SERBYT, même si cet octet a été « caché » dans SERBYT depuis le dernier épisode de communication série. Vous aurez affaire à ce « fripon d'octet » s'il apparaît au début d'une série d'octets reçus; tous les suivants seront encore reçus, mais leur sens peut avoir été altéré. Le moyen le plus simple pour se débarrasser de ce qui est dans SERBYT est de mettre SERFL à zéro avant de tenter de lire quoi que ce soit de l'interface série. Cela peut être fait avec les instructions machine suivantes :

```
XOR   A
LD    (23751),A
```

C'est la seule complication réelle que trouveront la plupart des utilisateurs de l'interface série — bien que, si l'on est incapable de s'en sortir, on puisse échanger les connexions aux broches 2 et 3 sur la prise d'Interface 1, au cas où elles auraient été mal connectées au départ. Le débit en bauds engendré, bien que pas toujours défini avec précision, est correct dans les tolérances requises par les spécifications RS232.

A bien des égards, la liaison RS232 est la plus facile à utiliser des fonctions de l'Interface 1. Il n'y a pas à s'occuper de canal d'information en code machine, et nous avons simplement deux appels à traiter : la transmission et la réception.

Moyens de communication
Le port série de l'Interface 1 fournit un moyen de communication avec d'autres appareils série et utilise une prise standard à neuf broches D. Le seul inconvénient de l'interface série Spectrum est son incapacité à utiliser différents débits pour la réception et la transmission.
(Cl. Kevin Jones.)

Pour débuter

Voici une sélection de livres pour vous initier à ces étranges et merveilleuses machines que sont les micro-ordinateurs. Ils vous feront entrer dans le cercle des amateurs éclairés.

Amstrad : programmation en assembleur

Toute application professionnelle sérieuse impose dans les faits le recours à l'assembleur ou au langage machine. Ces deux langages très voisins sont moins difficiles qu'on ne le croit, mais réclament une vive attention au détail.

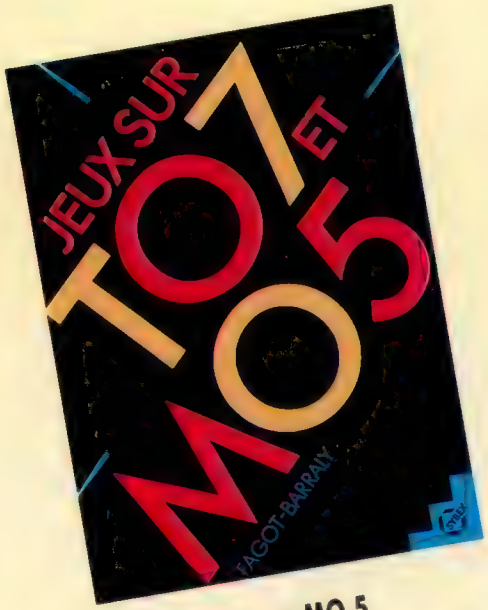
L'auteur donne une bonne présentation du langage assembleur, en partant de généralités exposées avec simplicité pour passer ensuite à une étude détaillée de chacune des instructions du microprocesseur Z80 qui équipe l'Amstrad.

Par Fagot-Barraly.
206 pages.
Sybex.

L'Amstrad exploré

L'auteur de ce livre donne d'abord une présentation, succincte mais précise, de l'appareil, puis aborde en détail ses possibilités graphiques et sonores, qui sont particulièrement riches. Vous apprendrez à créer des sons et à manipuler les différents modes graphiques. Une intéressante initiation au langage assembleur et un programme de comptabilité complètent heureusement un ouvrage délibérément destiné à des applications pratiques.

Par John Braga.
184 pages.
Sybex.

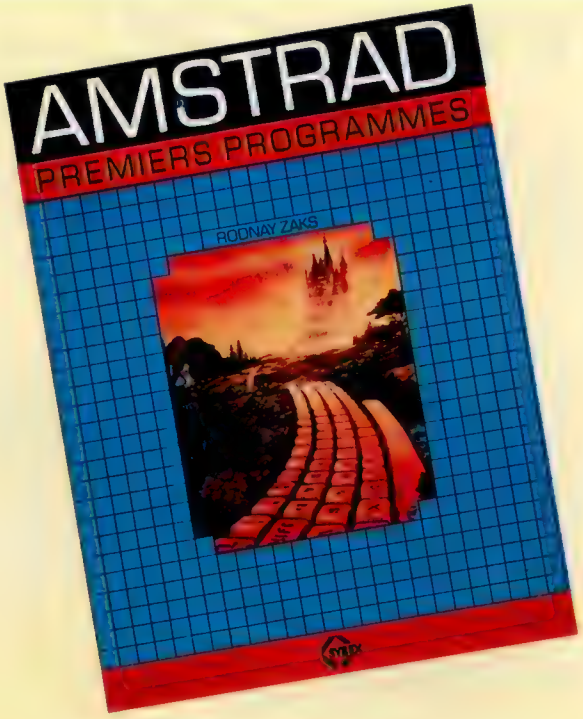


Jeux sur TO 7 et MO 5

Un recueil de listages : en tout quinze jeux d'arcades différents, avec des indications qui vous permettront de les modifier ou de les adapter sur d'autres appareils. Un bon moyen pour se familiariser avec la programmation sans jamais s'ennuyer, et pour apprendre à rédiger soi-même des programmes.

Par Fagot-Barraly.
206 pages.
Sybex.

Amstrad, premiers programmes



D'origine britannique, l'Amstrad CPC 464 connaît déjà un très grand succès en France, en raison de ses vastes capacités, mais aussi de son prix très intéressant. Cet ouvrage est avant tout une initiation très claire au BASIC de l'appareil, et passe sans efforts de la définition des instructions à la conception des programmes. Cette approche progressive, toujours accompagnée d'exemples commentés, permettra au lecteur d'aller du stade de la découverte à celui de la programmation avancée sans jamais négliger aucun détail, et de devenir à son tour un programmeur compétent.

Par Rodnay Zaks.
248 pages.
Sybex.

Guide du basic MSX

Les constructeurs japonais ont mis au point le standard MSX, qui assure la compatibilité entre ordinateurs différents : tout programme conçu pour l'un peut tourner sur tous les autres. On doit à la firme américaine Microsoft le BASIC, particulièrement élaboré, de ces appareils. Le livre de Michel Laurent est une étude très détaillée de toutes les instructions disponibles. Ce dictionnaire extrêmement complet est de surcroît enrichi de nombreux exemples.

Par Michel Laurent.
Sybex.

Gestion de fichiers sur TO 7 et MO 5

La gestion de fichiers est l'une des applications les plus intéressantes de la micro-informatique. Plutôt que d'en aborder l'étude de façon abstraite, l'auteur a choisi de partir d'un exemple précis : celui d'une gestion de bibliothèque. Il donne donc un programme très détaillé, qu'il décompose en modules dont chacun est disséqué et commenté à l'aide d'ordinogrammes.

Le lecteur disposera donc à la fois d'un utilitaire très pratique et d'une analyse lui permettant non seulement d'en comprendre la logique, mais également de le modifier selon ses besoins.

Par Jean-Pierre Lhoir.
136 pages.
Sybex.

Philips MSX

Dix-huit mois après avoir lancé son premier micro-ordinateur familial, Philips, en se ralliant au standard MSX avec les VG 8010 et 8020, est devenu le cheval de Troie des Japonais.

Peu avant Noël 1984, Philips annonçait à grand bruit son arrivée sur le marché de l'informatique familiale avec un petit ordinateur à la portée de tous, et surtout des jeunes, le VG 5000. Il lançait en même temps, dans plusieurs pays européens, le VG 8000, le premier micro-ordinateur européen au standard japonais MSX. Les Français pouvaient déplorer que, bien que plus coûteux, ce dernier ne leur soit pas destiné.

Avec l'apparition des deux nouveaux modèles, VG 8010 et VG 8020, voilà cette lacune comblée. Il semble que Philips, qui doutait d'abord de la pénétration des machines MSX en France, soit maintenant rassuré à ce sujet. En effet, depuis près de deux ans, les fabricants de logiciels n'ont cessé de développer des produits pour ces machines, essentiellement japonaises jusqu'à présent : Canon, Sanyo, Yamaha, Yashika, Yeno... Le marché français semble désormais prêt à accueillir les petits nouveaux de Philips.

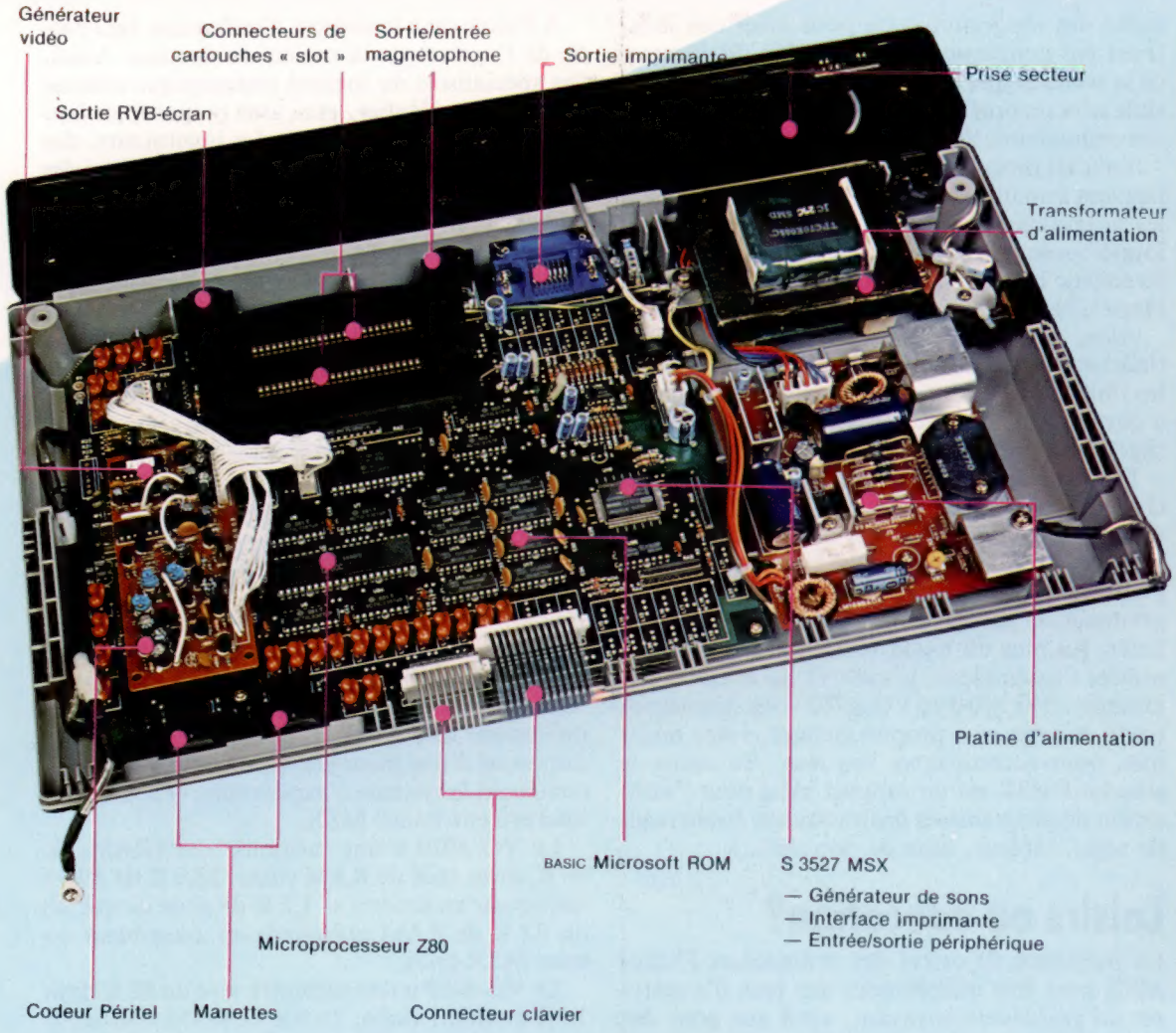
Un système évolutif

MSX *(Microsoft Super Extended)* est un standard permettant une compatibilité parfaite entre ordinateurs, périphériques et programmes. Vous pourrez donc, au rythme de vos besoins, faire évoluer votre configuration dans le temps à partir d'un micro MSX, sans vous soucier, par exemple, des problèmes de connexion de périphériques ou de compatibilité de programmes.

Les ordinateurs VG 8010 et 8020 de Philips bénéficient de ce standard. Simples d'utilisation et particulièrement performants, ils vous ouvriront le monde fascinant de la création, vous accompagneront dans votre découverte de la micro-informatique, et vous permettront, grâce à leur puissance, d'utiliser des programmes de plus en plus évolués.

Tout un choix de périphériques compatibles

De plus, grâce à la standardisation, vous pourrez utiliser tous les périphériques déjà créés pour MSX tels que lecteur de cassette, imprimante, lecteur de disquette, interfaces diverses, extension de mémoire, et une large gamme de programmes couvrant les domaines des loisirs, de l'éducation, de la création, de la programmation, de la gestion. Votre ordinateur Philips MSX deviendra vite un nouvel ami, fidèle, intelligent et sérieux. Tout le monde n'utilise pas un ordinateur pour



PHILIPS VG8010 MSX

DIMENSIONS

365 (L) × 242 (P) × 50 (H) mm.

UC

Z80, fonctionnant à 3,6 MHz.

MÉMOIRE

32 K ROM contenant le système d'exploitation et l'interpréteur BASIC MSX.
48 K RAM dont 16 K RAM vidéo.

ÉCRAN

Monochrome 31 cm, haute résolution, traité antireflet.
Phosphore vert (type P42);
Texte :
24 × 40 caractères.
Graphique :
256 × 192 points,
32 lutins.

INTERFACES

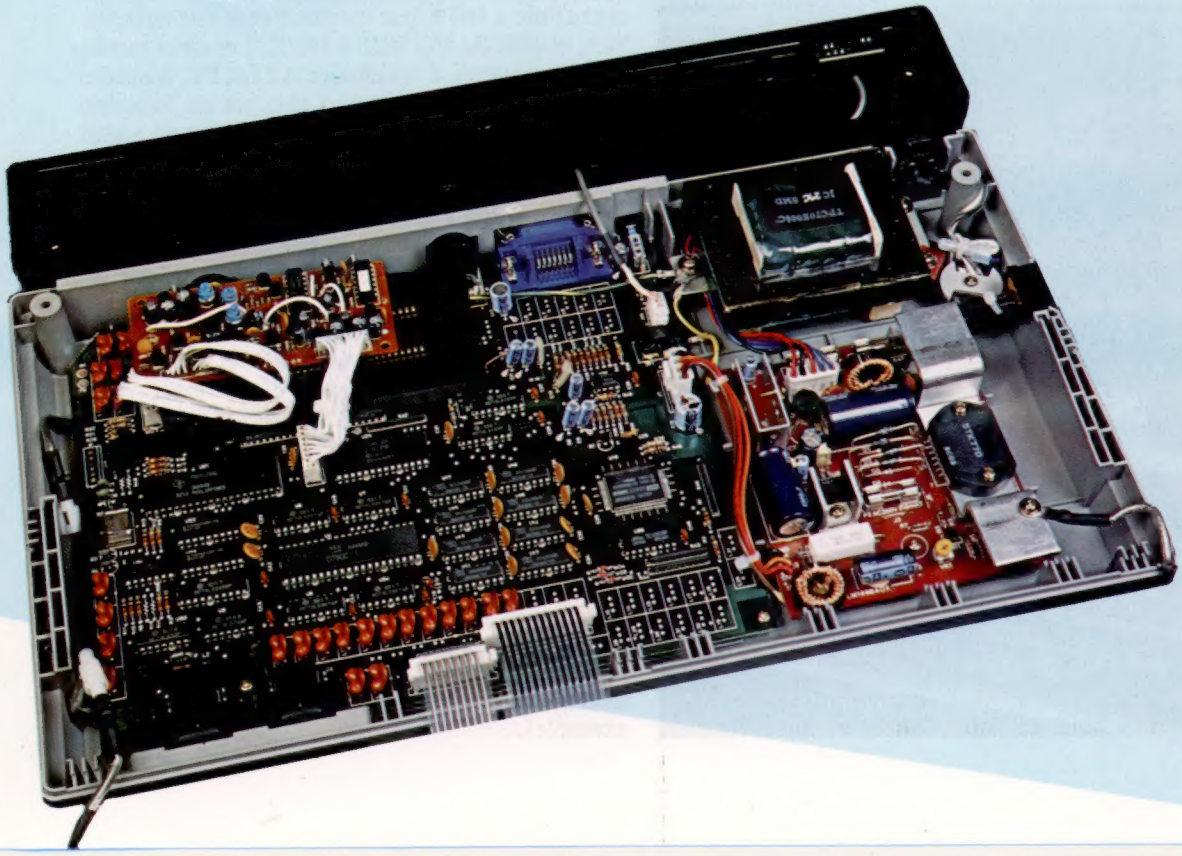
Sortie RVB et luminance-synchro.
Magnétophone à cassette.
Manettes de jeu.
2 slots pour cartouches ROM, RAM ou périphériques.

ALIMENTATION

Extérieure.

DOCUMENTATION

Manuel d'installation.
Manuel de référence BASIC MSX.



PHILIPS VG8020 MSX

DIMENSIONS

404 (L) × 220 (P) × 65 (H) mm.

UC

Z80, fonctionnant à 3,6 MHz.

MÉMOIRE

32 K ROM contenant le système d'exploitation et l'interpréteur BASIC MSX.
80 K RAM dont 16 K RAM vidéo.

ÉCRAN

Le même que pour le VG8010.

INTERFACES

Les mêmes que pour le VG8010 et, en plus, interface imprimante.

ALIMENTATION

Intégrée.

DOCUMENTATION

Manuel d'installation.
Manuel BASIC MSX.

écrire des programmes ou pour créer des jeux. Il est par conséquent indispensable de disposer de la bibliothèque de logiciels la plus étendue possible afin de profiter pleinement des possibilités des ordinateurs VG 8010 et VG 8020 de Philips.

Tous les programmes portant la mention MSX peuvent être utilisés sur ces modèles, que ce soit les programmes Philips ou ceux qui sont développés pour MSX par des éditeurs français et étrangers comme Activision, Ascii Microsoft, Hatier, Infogrammes, Loriciels, Vifi Nathan...

Alors que le petit frère VG 5000 était essentiellement destiné à l'initiation à l'informatique, les Philips MSX sont des ordinateurs personnels à part entière, destinés aussi bien à l'apprentissage qu'au jeu ou à la gestion professionnelle.

Des logiciels très diversifiés sur support cartouche, cassette ou disquette vous permettront d'apprendre : le VG 8010 et le VG 8020 sont en effet des ordinateurs particulièrement adaptés à l'éducation. Pour jouer, la bibliothèque de programmes de jeux pour MSX suffira à vous rassurer. En plus du BASIC résident, vous pourrez utiliser l'assembleur, le PASCAL ou le LOGO par exemple. VG 8010 et VG 8020 vous donneront envie de créer vos propres images, votre musique, pour accompagner vos jeux. En outre, le standard MSX est un support idéal pour l'utilisation de programmes professionnels (traitement de texte, tableur, base de données...).

Loisirs ou éducation?

La puissance de calcul des ordinateurs Philips MSX peut être utilisée pour des jeux d'aventures au graphisme attrayant, ainsi que pour des jeux d'adresse et de réflexes. Vous pourrez aussi vous passionner pour une partie d'échecs ou utiliser un programme de simulation de vol.

A l'école ou à la maison, l'ordinateur fera partie de l'univers de la société de demain. Aussi, des spécialistes du logiciel pédagogique comme Vifi Nathan, Hatier, etc., ont préparé, en liaison avec des professeurs et des instituteurs, des programmes éducatifs au standard MSX dans des domaines qui couvrent l'apprentissage de la programmation, les mathématiques, l'orthographe, les langues...

Pour vous permettre de tirer le meilleur parti de votre micro-ordinateur, les unités centrales VG 8010 et VG 8020 MSX sont livrées avec une documentation complète indiquant les consignes d'installation et toutes les possibilités d'utilisation du BASIC MSX. Chacune des 130 instructions est décrite, expliquée et illustrée d'exemples d'utilisations.

Des caractéristiques intéressantes

Comme tous les micros MSX, les machines Philips VG 8010 et VG 8020 présentent les caractéristiques suivantes : elles utilisent un microprocesseur Z80 fonctionnant à 3,6 MHz. Elles disposent d'une mémoire morte (ROM) de 32 K contenant le système d'exploitation et le puissant interpréteur BASIC MSX.

Le VG 8010 a une mémoire vive (RAM) de 48 K, dont 16 K de RAM vidéo, 28,8 K de RAM utilisateur en BASIC (+ 3,2 K de zone de calcul) ou 32 K de RAM utilisateur en assembleur ou sous MSX-DOS.

Le VG 8020 a une mémoire vive de 80 K dont 16 K de RAM vidéo, 28,8 K de RAM utilisateur en BASIC (+ 3,2 K de zone de calcul) ou 64 K de RAM utilisateur en assembleur ou sous MSX-DOS.

La mémoire vive utilisateur du VG 8010 est extensible à 160 K par cartouches RAM enfichables, et celle du VG 8020 à 192 K. Les deux modèles sont équipés d'un clavier AZERTY, 6 modes (avec lettres accentuées), 5 touches de fonction permettant la sélection de 10 instructions BASIC programmables à volonté.

Le puissant microprocesseur vidéo permet de définir jusqu'à 256 lutins, dont 32 peuvent s'afficher simultanément. Les écrans disposent de 16 couleurs et d'une résolution graphique de 256 × 192 points.

Les polices de caractères utilisées comportent 253 caractères alphanumériques et graphiques différents. Formés à partir d'une matrice de 8 × 8 points, ils peuvent s'afficher en mode texte (24 lignes de 40 caractères) ou en mode graphique (24 lignes de 32 caractères).

Le générateur de sons du VG 8010 et du VG 8020 permet de reproduire simultanément 3 sons sur 8 octaves.

Les micro-ordinateurs Philips MSX possèdent deux « slots » destinés à recevoir des logiciels sous forme de cartouches ROM enfichables, des cartouches d'extension de mémoire vive, ou des interfaces périphériques (lecteur de disquette, par exemple).

Page manquante
(publicité)

Page manquante
(publicité)